

1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2752 2753 2754 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768 2769 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806 2807 2808

TO ALL WHOM IT MAY CONCERN:

# System and Methods for Encrypted Execution of Computer Programs

of which the following is a specification.

## System and Methods for Encrypted Execution of Computer Programs

### Related Applications

This application claims the benefit of U.S. Provisional Application No. 60/181,363, filed February 8, 2000, entitled System and Methods for Executed Program Encryption, which is expressly incorporated herein by reference.

### Background of the Invention

Distributed network-based computing is growing quickly, particularly with the computerization of corporations and the expansion of the Internet. Security for distributed network-based computing is thus of utmost importance to ensure data integrity and privacy, and to maintain corporate and intellectual integrity.

One particular threat to security in distributed network-based computing involves program execution on a remote host ("host" is used herein synonymously with host computer) that may or may not be reliable or "trusted." The prior art has been unable to ensure that (a) such program execution is performed correctly or (b) the remote host learned nothing of computations or data associated with underlying program execution. Accordingly, improvements to network security to address these security issues are needed.

One object of the invention is to provide a method for encrypting and subsequently executing programs on a remote computer in a manner that solves the afore-mentioned problems. Special computing hardware is not required. This object and others will be apparent in the description that follows.

### Summary of the Invention

In one aspect, the invention provides a method for encrypting a program for execution on a remote host computer on a network, such that incorrect execution by the remote host computer is detectable, and such that the underlying software code remains secure, and such

1nsa' >

00754604300

that the computations and the data associated with execution are unintelligible standing alone and otherwise useless at the host computer. Results from the encrypted computations at the remote host computer are transmitted to a control computer, on the network, which decodes the results to useful data representative as output from the program. In a first step of the method, the program is encoded as a unitary matrix,  $U=(U_{ij})$ , with  $n$  rows and  $n$  columns, such that the execution of the program on a dataset is realized by matrix multiplication of the matrix  $U$  with the vector of input data bits.

This encoding is a representation of computer programs expressed in terms of reversible logic or quantum gates. That such a representation is possible can be illustrated as follows. All operations of a Turing computer can be reduced to the primitive logic operations, "XOR" (exclusive OR) and "NOT" (negation), operating appropriately on binary bits of data. For example, if  $r$  and  $s$  are two bits, then " $r$  XOR  $s$ " is the bit 1 (true) if either but not both  $r$  and  $s$  are 1 while " $r$  XOR  $s$ " is 0 (false) otherwise. Similarly, "NOT  $r$ " is 0 if  $r$  is 1 and 1 if  $r$  is 0. These two operations constitute binary operations or binary logic on which digital computers are based. The basic operation of digital computers can be reduced to these two primitive operations combined in a circuit. Moreover, computer programs operating on input data can be expressed as a predetermined sequence of such binary logic operations. If the program consists of a finite number of steps, then the program can be expressed as a predetermined finite sequence of binary logic operations.

The representation of a computer program using such binary logic operations is in general not "reversible." Evidently, if " $r$  XOR  $s$ " has a value of 0 for example, the values of  $r$  and  $s$  individually cannot be determined because they could be any one of the cases:  $r=s=0$ ;  $r=s=1$ . The operation "NOT" is reversible. Reversible logic represents a bit as a 2-dimensional vector,  $(1\ 0)'$  for 1 (true) and  $(0\ 1)'$  for 0 (false). In this specification, the notation  $'$  denotes the transpose of a matrix or a vector. Two bits are then two 2-dimensional vectors that can be represented in a combined way by a 4-dimensional vector. The possible values of such a four dimensional representation are  $a=(1\ 0\ 0\ 0)'$ ,  $b=(0\ 1\ 0\ 0)'$ ,  $c=(0\ 0\ 1\ 0)'$  and  $d=(0\ 0\ 0\ 1)'$  where  $a=(1\ 0\ 0\ 0)'$  represents input bits both false,  $b=(0\ 1\ 0\ 0)'$  represents input bit 1 false and input bit 2 true,  $c=(0\ 0\ 1\ 0)'$  represents input bit 1 true and input bit 2

false and  $d=(0\ 0\ 0\ 1)'$  represents both input bits being true. The same representation can be used for the output bits with the convention that the first bit of the output represents the logical value of the XOR operation and the second bit is used for reversibility.

For example, both  $b$  and  $d$  are outputs with XOR value true. We can define the XOR logic by the following mapping from inputs to outputs:  $a$  goes to  $a$ ,  $b$  goes to  $c$ ,  $c$  goes to  $d$  and  $d$  goes to  $b$ . In matrix-vector multiplication terms, this can be represented by the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

which is a unitary matrix, verified by multiplying the matrix by its transpose. Similarly, the NOT operator can be represented as the unitary

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

which can be verified by direct multiplication. These two constructions demonstrate that elementary logic operations can be represented as unitary operations. Since the direct product and composition of unitary operations is a unitary operation, this shows that a finite number of steps of a Turing program can be represented as a unitary matrix operation.

The set of unitary matrices with  $n$  rows and  $n$  columns, denoted by  $U_n$ , forms a non-commutative group under standard matrix multiplication. It therefore has a unique group-invariant Haar measure that determines an intrinsic uniform probability distribution. The program encoding,  $U$ , is thus an element of a group,  $U_n$ , having an intrinsic uniform Haar probability distribution. In a second step, an input data string to the program is encoded as a vector,  $b=(b_j)$ , of length  $n$ . The first and second steps can be performed in either order or simultaneously. In a third step, two independent identically distributed unitary matrices,  $X=(X_{ij})$  and  $Y=(Y_{ij})$ , are generated from the uniform probability distribution over  $U_n$  determined by the Haar distribution.  $X$  and  $Y$  are therefore uniformly randomly distributed within the group when generated in this way. In a fourth step,  $U'$  is computed as  $XUY^*$  and

sent to the remote host over the network. In a fifth step,  $b'$  is computed as  $Yb$  and sent to the remote host over the network. The fourth and fifth steps can be performed in either order. In a sixth step, the remote host computes the product of  $XUY^*$  and  $Yb$  and sends the result to the control computer on the network. In a seventh step, the control computer computes  $X^*XUb$  to determine the multiplication of  $Ub$ , the desired output of the program.

The prior art is familiar with Alan Turing, the British mathematician who in 1936 published a paper called "On Computable Numbers with an Application to the *Entscheidungsproblem*". It is generally recognized by those skilled in the art that Turing's 1936 paper founded modern theories of computation and introduced the concept of a "Turing Machine." The Turing Machine is said to perform a computation (a "Turing computation") using a tape of infinite length with an infinite series of square cells. In one step of its operation, a Turing Machine device reads a symbol from the tape, alters its own internal state, possibly changes the symbol in the tape cell it just read and then moves the tape left or right to another cell tape. Another step is then performed. The afore-mentioned method of the invention relies, in part, on the premise that matrix multiplication, specifically unitary matrix multiplication, is a general representation for a computer program determining a fixed number of steps of a Turing computation. Moreover, the method of the invention relies further on the fact that the execution of such a Turing computation on the input data string that is initially on the tape is entirely equivalent to the matrix product of the unitary matrix,  $U$ , representing the Turing machine computation or program, with an input data vector,  $b$ . The product is hereinabove described as  $Ub$ .

The encryption scheme of the invention, described in more detail below, has specific advantages stemming from the uniform distribution of unitary matrices. Specifically, encryption of programs according to the invention in terms of random unitary matrix pre- and post-multiplications results in Haar uniform probabilistic distribution of the encrypted programs – wherein any two programs or data strings of the same size that are encrypted separately according to this method will have the identical statistical distribution of all data in their encoded representations. Since all programs of a fixed size will have encrypted representations that are statistically indistinguishable, the remote computer will not be able to

learn anything about a specific program because there is nothing that distinguishes any one such encrypted program from any other such encrypted program.

In another aspect, the method has the further step of embedding constants into the data and program prior to the step of encryption, to enable detection of incorrect execution at the remote host or tampering with either the data or program representations. This can be

accomplished by appending a trailing 1, as  $\begin{bmatrix} b \\ 1 \end{bmatrix}$ , to the input data vector  $b$  and embedding  $U$

into an  $(n+1)$  by  $(n+1)$  unitary as  $\begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix}$  so that the correct output is  $\begin{bmatrix} Ub \\ 1 \end{bmatrix}$ . Any change to

$X \begin{bmatrix} Ub \\ 1 \end{bmatrix}$  either through error or malicious tampering will result in a decoded answer that will

not have exactly 1 as the trailing entry of the decoded output vector  $\begin{bmatrix} Ub \\ 1 \end{bmatrix}$ .

These and other aspects will become more apparent in reviewing the description that follows.

#### Brief Description of the Drawings

**FIG. 1** shows a distributed network computing system constructed according to the invention; and

**FIG. 2** illustrates a flow chart for steps taken in the system of **FIG. 1**.

#### Detailed Description of the Drawings

**FIG. 1** shows a distributed network computing system 10 constructed according to the invention. System 10 includes a control computer 12 and a remote host computer 16, connected together by computing network 14. Other computers 18 can be, and usually are, connected to network 14 without departing from the scope of the invention. Network 14 can

be the Internet, a virtual private network, LAN or other network, wired or wireless, connecting computing devices together.

In operation, a program to be executed on host 16 is first encrypted on control computer 12 and sent to host 16 over network 14. Data for input to the program is also encrypted at control computer 12 or elsewhere 18 in the network 14 and sent to host 16 over network 14. Host 16 then executes an encrypted form of the program using the encrypted form of the data; and transmits results through network 14. Control computer 12 (or another computer with the decode information) then accesses and decodes the results to determine the desired output.

These operations occur through matrix multiplication. As described herein, this means that the program is first converted to a matrix of the form  $A = (a_{ij})$ , and that the input data for the program is converted to a vector of the form  $b = (b_j)$ , such that  $A$  and  $b$  are compatible for multiplication. The result describing the desired output from executing the program with the data string is a matrix-vector product  $c = Ab$ , defined as

$$c_i = \sum_j a_{ij} b_j$$

$A$  and  $b$  are next encrypted so that the details of  $A$  and  $b$ , and therefore also of  $c$ , are not known at host computer 16. Specifically,  $A$  is converted to  $A' = XAY^T$  and  $b$  is converted to  $b' = Yb$ , where  $X$  and  $Y$  are matrices of the same size as  $A$  that have been independently generated as random unitary matrices with respect to the Haar distribution on the group of unitary matrices.  $A'$  and  $b'$  are then sent to host computer 16 for execution. Host computer 16 computes  $c' = A'b' = (XAY^T)(Yb) = XAb$ . Subsequently  $XAb$ , sent back to the control computer 12, is decoded by computing  $X^T A' b' = Ab = c$ , the desired result.

Matrices  $A$ ,  $b$  are unitary matrix multiplications. Unitary operators or matrices are square matrices,  $P$ , defined by the property that  $PP^* = P^*P = I$ , where  $I$  is the identity matrix. Another way of stating this property is to say that for unitary  $P$ ,  $P^* = P^{-1}$ , the

inverse matrix of  $P$ . The matrix  $P^*$  is the complex conjugate transpose of  $P$  so that if the  $ij$ -th entry of  $P$  is  $P_{ij} = a_{ij} + i b_{ij}$  then the  $ij$ -th entry of  $P^*$  is  $a_{ji} - i b_{ji}$  where  $a_{ji}$  and  $b_{ji}$  are real numbers. Therefore, unitary operators are invertible and their inverses are equal to their complex conjugate transpose:  $P^{-1} = P^*$ . By ensuring that the data string and program are encoded, respectively, as a vector and a unitary operator, then the multiplication of the vector by the unitary operator defines steps of Turing computations to determine the desired output. Specifically, the unitary operator encodes the finite Turing computation and the vector encodes the input data string.

The above unitary operators are also randomized. Since unitary matrices are non-commutative under matrix multiplication, they each have unique group-invariant probability distributions (sometimes denoted as the Haar measure). If  $U_n$  denotes the unitary operators on complex  $n$ -space,  $C_n$ , and  $S \subset U_n$  is a measurable subset of  $U_n$  then the Haar probability distribution over  $U_n$  satisfies:

$$1 \geq \mu(US) = \mu(SU) = \mu(S) \geq 0$$

for all  $U \in U_n$ . That is, the probability of a set is invariant under the action of multiplication by a group element. In addition,  $\mu(U_n) = 1$ . Accordingly, the Haar distribution is the intrinsic uniform probability distribution of  $U_n$ . If  $X$  is a random unitary matrix distributed according to the Haar distribution, and  $U$  is a fixed unitary, then the products  $XU$  and  $UX$  are uniformly distributed according to the Haar distribution as well.

An important part of the above encryption, therefore, is the effective generation of unitary matrices, distributed according to the Haar probability distribution on  $U_n$ . G.W. Stewart, in "The efficient generation of random orthogonal matrices with an application to condition estimators," SIAM J. Numer. Anal., 17:403-409 (1980), incorporated herein by reference, provides one technique for generating random unitary matrices.

**FIG. 2** is a flow chart 50 illustrating steps occurring between host computer 16 and control computer 12. Those skilled in the art should appreciate that control computer 12 is



illustrative in that like functions of control computer 12 may be performed on other computers connected to network 14; alternatively, like functions of control computer 12 may be distributed among several other computers connected to network 14. The important feature of encryption as described is that host computer 16 does not glean intelligible or otherwise useful information of the underlying program or program software code, or of the data and computations entered as input or generated thereby.

Processing in a method 50 in accordance with the invention, as depicted in **FIG. 2**, begins at control computer 12 (or other computer in network with host computer 16), identified as step 52. Step 54 and 56 may be done concurrently or in either order: step 54 converts a program to a unitary matrix multiplication  $U \in U_n$  for some integer  $n$ ; step 56 converts a data string, for input to the program, to a vector  $b$ . In step 58, two independent identically distributed unitary matrices, e.g.,  $X$ ,  $Y$ , are generated from the Haar distribution over  $U_n$ . Step 60 and 62 may be done concurrently or in either order: step 60 computes  $U'$  as  $XUY^*$  and sends  $U'$  to the remote host over the network; step 62 computes  $b'$  as  $Yb$  and sends  $b'$  to the remote host over the network. In step 64, remote host 16 computes the product of  $XUY^*$  and  $Yb$  and sends the results to control computer 12 on network 14. In step 66, control computer 12 computes  $X^*XUb$  to determine the multiplication of  $Ub$ , the desired output of the program. Processing can continue for other programs or data, or alternatively end at step 68.

The invention thus attains the objects set forth above, among those apparent from preceding description. Since certain changes may be made in the above systems and methods without departing from the scope of the invention, it is intended that all matter contained in the above description or shown in the accompanying drawing be interpreted as illustrative and not in a limiting sense.

In view of the foregoing, what is claimed is: